Homework 2

February 1, 2025

1 Show the following relations are correct:

1.1 $(x+1)^n = 1 + nx + o(x)$ **as** $x \to 0$

$$f(x) = o(x) = (x+1)^n - 1 - nx \text{ and } g(x) = x$$
$$\lim_{x\to 0} \frac{(x+1)^n - 1 - nx}{x} = \frac{0}{0} \text{ (L'Hopital)}$$
$$\lim_{x\to 0} \frac{n(x+1)^{n-1} - n}{1} = 0$$

Thus, the error in the Taylor Series goes to 0 like $o(x)$.

1.2 $x \sin\sqrt{x} = O(x^{3/2})$ **as** $x \to 0$

$$\sin x = x - \frac{x^3}{6} + \frac{x^5}{24} + ...$$
$$\sin\sqrt{x} = \sqrt{x} - \frac{\sqrt{x}^3}{6} + \frac{\sqrt{x}^5}{24} + ...$$
$$x\sin\sqrt{x} = x\sqrt{x} - x\frac{\sqrt{x}^3}{6} + x\frac{\sqrt{x}^5}{24} + ...$$
$$x\sin\sqrt{x} = x^{3/2} - \frac{x^{5/2}}{6} + \frac{x^{7/2}}{24} + ...$$
$$g(x) = x^{3/2},\ f(x) = x\sin\sqrt{x}$$
$$\lim_{x\to 0} \frac{x\sin\sqrt{x}}{x^{3/2}} = \frac{0}{0} \text{ (L'Hopital)}$$
$$\lim_{x\to 0} \frac{x^{3/2} - \frac{x^{5/2}}{6} + \frac{x^{7/2}}{24} + ...}{x^{3/2}} = \lim_{x\to 0}(1 + \frac{-\frac{x^{5/2}}{6} + \frac{x^{7/2}}{24} + ...}{x^{3/2}}) = 1$$

This limit exists and is a nonzero, so the Taylor series error goes to zero at least as fast as $x^{3/2}$.

1.3 $e^{-t} = o(\frac{1}{t^2})$ **as** $t \to \infty$

$$g(t) = \frac{1}{t^2} \text{ and } f(t) = e^{-t}$$
$$\lim_{t\to 0} \frac{e^{-t}}{1/t^2} = \lim_{t\to 0} t^2 e^{-t} = 0$$

Thus, the error in the Taylor Series goes to 0 like $o(1/t^2)$.

1.4 $\int_0^\varepsilon e^{-x^2} dx = O(\varepsilon)$ **as** $\varepsilon \to 0$

$$g(\varepsilon) = \varepsilon \text{ and } f(\varepsilon) = \int_0^\varepsilon e^{-x^2} dx$$
$$\lim_{\varepsilon\to 0} \frac{\int_0^\varepsilon e^{-x^2} dx}{\varepsilon} = \frac{0}{0} \text{ (L'Hopital)}$$
$$\lim_{\varepsilon\to 0} \frac{e^{\varepsilon^2}}{1} = 1$$

This limit exists and is a nonzero, so the Taylor series error goes to zero at least as fast as ε

2 Consider solving Ax=b by investigating perturbations of b and condition number.

$A = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1+10^{-10} & 1-10^{-10} \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $A^{-1} = \begin{bmatrix} 1-10^{10} & 10^{10} \\ 1+10^{10} & -10^{10} \end{bmatrix}$ with perturbation: $\mathbf{\Delta b} = \begin{bmatrix} \Delta b_1 \\ \Delta b_2 \end{bmatrix}$

2.1 Find an exact formula for the change in the solution between the exact problem and the perturbed problem Δx.

$$A\mathbf{x} = \mathbf{b}$$
$$\mathbf{x} = A^{-1}\mathbf{b}$$
$$\tilde{\mathbf{x}} = A^{-1}\tilde{\mathbf{b}}, \text{ where } \tilde{\mathbf{b}} = \begin{bmatrix} b_1 + \Delta b_1 \\ b_2 + \Delta b_2 \end{bmatrix}$$
$$\Delta\mathbf{x} = \tilde{\mathbf{x}} - \mathbf{x}$$
$$\Delta\mathbf{x} = A^{-1}\tilde{\mathbf{b}} - A^{-1}\mathbf{b}$$
$$\Delta\mathbf{x} = A^{-1}(\tilde{\mathbf{b}} - \mathbf{b})$$
$$\Delta\mathbf{x} = A^{-1}\begin{bmatrix} \Delta b_1 \\ \Delta b_2 \end{bmatrix}$$
$$\Delta\mathbf{x} = \begin{bmatrix} 1-10^{10} & 10^{10} \\ 1+10^{10} & -10^{10} \end{bmatrix}\begin{bmatrix} \Delta b_1 \\ \Delta b_2 \end{bmatrix}$$
$$\boxed{\Delta\mathbf{x} = \begin{bmatrix} (1-10^{10})\Delta b_1 + 10^{10}\Delta b_2 \\ (1+10^{10})\Delta b_1 - 10^{10}\Delta b_2 \end{bmatrix}}$$

2.2 What is the condition number of A?

$$\kappa_A = \frac{|\lambda_{max}|}{|\lambda_{min}|}$$

From Python program, the eigenvalues of A are $\lambda_1 = 1, \lambda_2 = -5.00000041 \times 10^{-11}$

$$\kappa_A = \frac{1}{|-5.00000041 \times 10^{-11}|}$$
$$\boxed{\kappa_A = 19999998345.19272}$$

This is a very large condition number so the problem and formula are poorly conditioned. Small disturbances in Δb have profound impacts on Δx.

2.3 Let $\Delta b_1.\Delta b_2$ be of magnitude 10^{-5} not necessarily the same value. What is the relative error in the solution? What is the relationship between the relative error, the condition number, and the perturbation. Is the behavior different if the perturbations are the same? Which is more realistic: same value of perturbation or different value of perturbation?

For any small perturbation in $\mathbf{b}$ there is a large perturbation in the output due the high condition number.

$$\mathbf{b} = \begin{bmatrix} 1.54 \times 10^{-5} \\ 8.71 \times 10^{-5} \end{bmatrix} \Rightarrow \Delta\mathbf{x} = \begin{bmatrix} 717001.0 \\ -716998.9 \end{bmatrix}$$

For no perturbation, $\mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$$\varepsilon_{rel} = \frac{\sqrt{2} - ||\Delta\mathbf{x}||}{\sqrt{2}}$$
$$\boxed{\varepsilon_{rel} = 716998.99999879}$$

Because of how large the error is in this system, the small disturbances in $\mathbf{b}$ lead to huge errors in the output. It is accurate that there would be different values for the disturbances. The larger the condition number, the larger the error in the output.

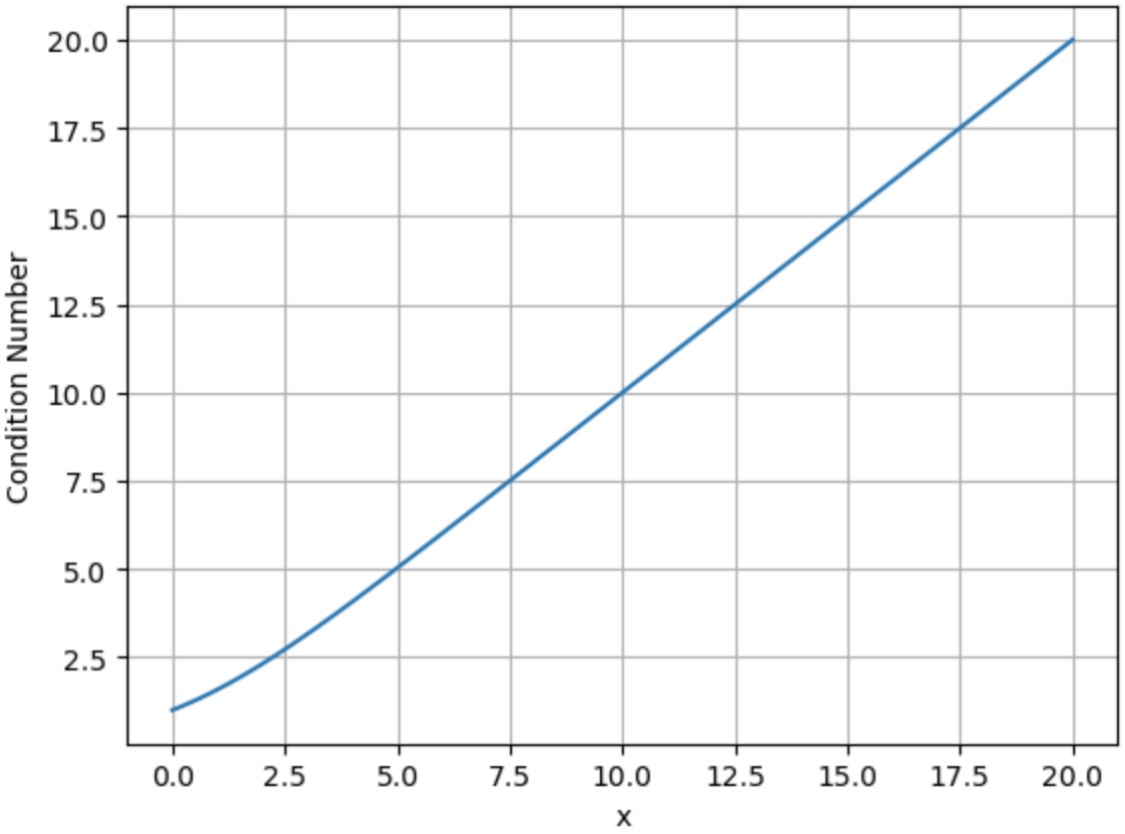
3 Let $f(x) = e^x - 1$

3.1 What is the relative condition number $\kappa(f(x))$? Are there any values of x for which this is ill conditioned?

For $x \approx 0$ this system will be ill conditioned, because there is subtraction of near numbers which leads to cancellation.

$$\kappa_f(x) = \frac{|x|}{|e^x - 1|}|e^x$$

The condition number shows the issue with $x \approx 0$ because the value of $\kappa_f(x)$ diverges due to the value of the denominator.This leads to a whole in the graph since this can be canceled out and the value is stil small. However, any values near 0 will diverge.



3.2 Consider computing $f(x)$ with the following algorithm. Is it stable? Justify.

y=math.e x̂
return y -1

For x not near 0, this is a fairly stable system. It increases in instability as x increases, because larger errors in the value of x lead to larger errors in the ouput $f(x)$. It is unstable at $x = 0$ because of the hole.

3.3 Let x be $9.999999995000000 \times 10^{-10}$, where $f(x) = 10^{-9}$ up to 16 decimal places. How many correct digits does this algorithm give? Is this expected?

The output is $f(x) = 1.000000082740371e - 09$. This is accurate to 7 decimals. This is expected. Since x is very small, it begins to cause the denominator to diverge the system. The log of the condition number represents the lost precision, so $\log 10^{-9} = -9$, meaning that 9 digits of decimal precision are lost. Then it makes sense that there are 7 digits of precision left over, since $7 = 16 - 9$.

3.4 Find a polynomial approximation of $f(x)$ that is accurate to 16 digits for x above. Hint: use Taylor series, and remember that 16 digits of accuracy is a relative error, not an absolute one.

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + ...$$
$$f(x) = e^x - 1 = x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + ...$$

From Taylor's Remainder Theorem,

$$10^{-16} \geq \frac{x^{n+1}e^z}{(n+1)!}$$

The first n that satisfies this is $n = 2$, so

$$f(x) \approx x + \frac{x^2}{2}$$

Using this approximation,

$$f(x) = 1e - 9$$

3.5 Verify that pard 3.4 is correct.

As shown above, implementing this method gives the expected result, $f(x) = 1e - 9$. See my Python Code for the implementation of this method.

3.6 Optional: How many digits of precision do you have if you do a simpler Taylor Series?

If you do a simpler Taylor series, you get $f(x) = x$, so $f(x) = 9.999999995000000e - 9$. This is not very accurate compared to the second order method, since no decimals are accurate. It is more accurate to calculate this quantity using the algorighm described above.

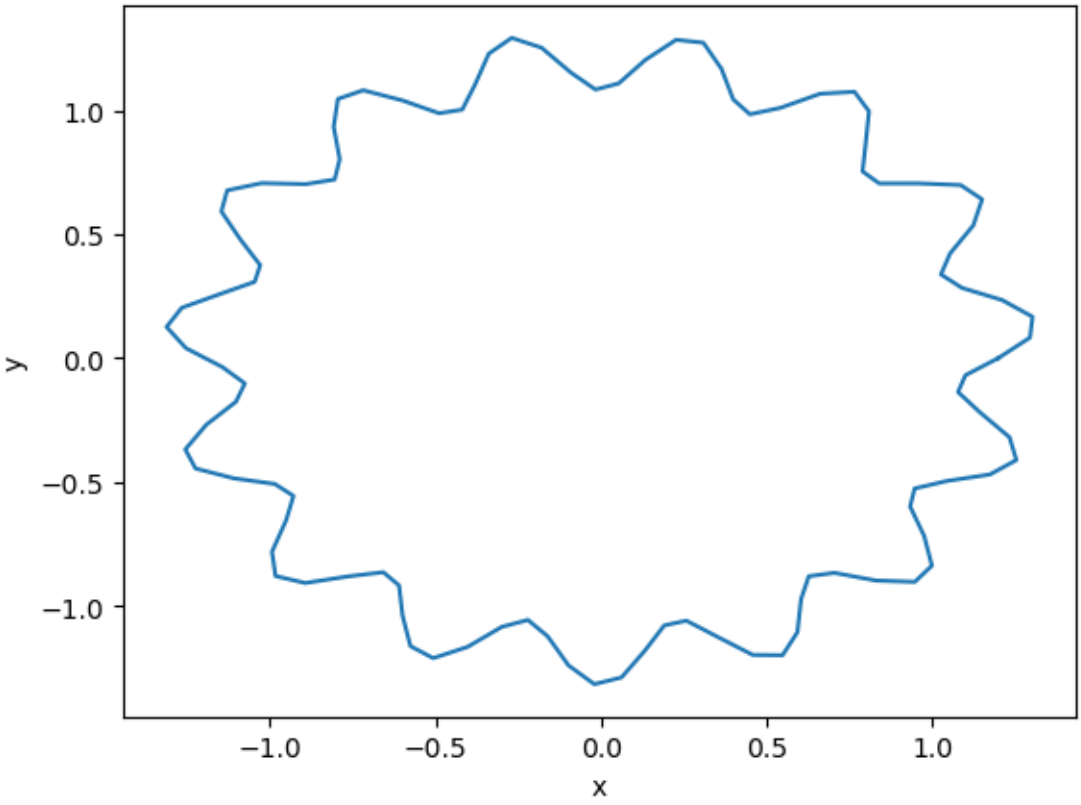
4 Practicing Python:

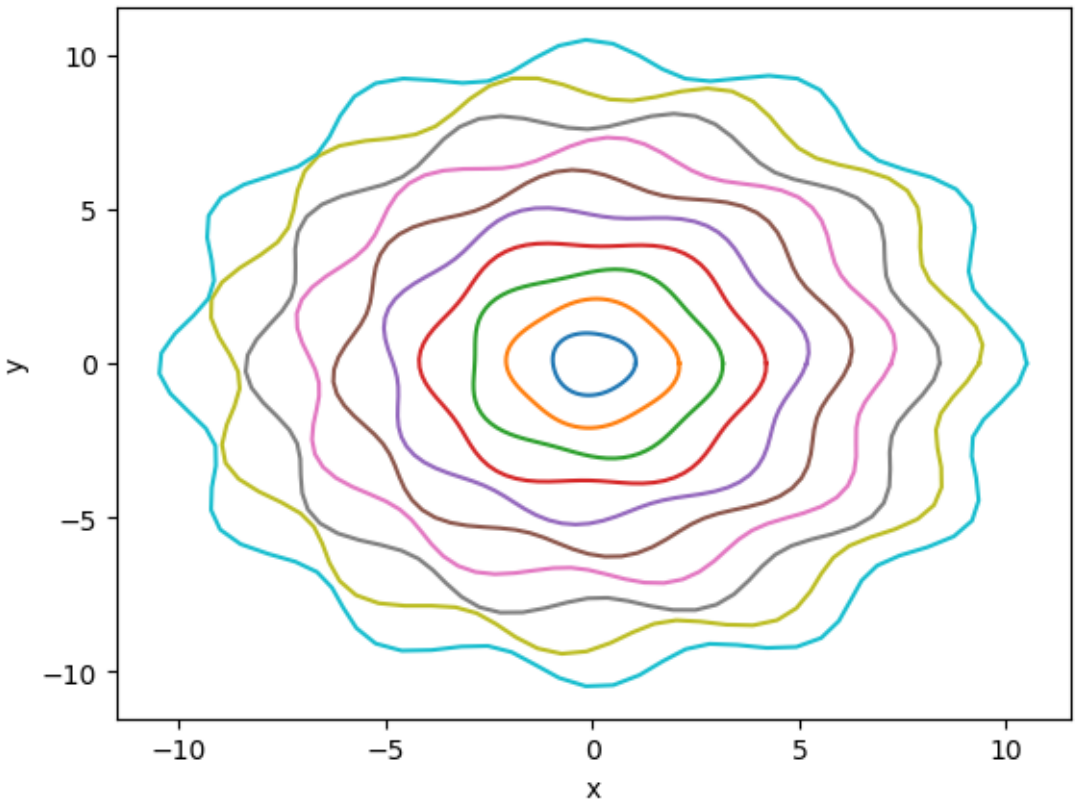
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import random

# Python practice a
# Create a vector t from 0:pi/30:pi, and create a vector y = cos(t). Evaluate the sum.

t = np.arange(0, np.pi + np.pi/30, np.pi/30)[:-1]

y = np.cos(t)
s = t*y
S = np.sum(s)
print('The sum is: ',S)
```

```
The sum is:  -20.68685236434684
```

[3]:
```
# Plotting

# Constants for the first figure
theta = np.linspace(0,2*np.pi,100)
R = 1.2
dr = 0.1
f = 15
p = 0

# Calculating x, y
x = R*(1+dr*np.sin(f*theta + p))*np.cos(theta)
y = R*(1+dr*np.sin(f*theta + p))*np.sin(theta)

# Creating the figure
plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('y')

# Creating the several graphs for i in [1,10]
x1 = []
x2 = []

i = np.linspace(1,10,10)

plt.figure()
plt.xlabel('x')
plt.ylabel('y')

dr = 0.05

# Iterating through all of the values and plotting
for i in i:
    R = i
    f = 2+i
    p = random.uniform(0,2)
```

```
a = R*(1+dr*np.sin(f*theta + p))*np.cos(theta)
b = R*(1+dr*np.sin(f*theta + p))*np.sin(theta)

np.append(x1,a)
np.append(x1,b)
plt.plot(a,b)
```




10
5
0
−5
−10
y
−10
−5
0
5
10
x